Adapting *word2vec* to Named Entity Recognition

Scharolta Katharina Sienčnik
Department of Swedish /
Department of Philosophy,
Linguistics and Theory of Science
University of Gothenburg, Sweden
gussieka@student.gu.se

Abstract

In this paper we explore how word vectors built using *word2vec* can be used to improve the performance of a classifier during Named Entity Recognition. Thereby, we discuss the best integration of word embeddings into the classification problem and consider the effect of the size of the unlabelled dataset on performance, reaching the unexpected result that for this particular task increasing the amount of unlabelled data does not necessarily increase the performance of the classifier.

1 Introduction

Supervised NLP systems suffer from a fundamental data bottleneck problem: though unprecedented amounts of data and the computational power necessary for its processing have become available, supervised training requires data that has been annotated for a specific task. The process of annotation in turn requires human time and is thereby inherently connected to high costs, both in terms of time and money.

Enhancing supervised methods with unsupervised word representations can ameliorate this problem. Word representations can be trained on large unannotated corpora and can learn implicit semantic and/or syntactic information. This information can then be used to augment a small amount of annotated data, thereby reducing the amount of annotated data necessary or improving the accuracy of a classifier with a given amount of annotated data.

Different word representations have been shown to successfully improve various NLP tasks. For example, Miller et al. (2004) use word clusters during named entity recognition (henceforth NER) and Bansal et al. (2014) use continuous word representations as features for dependency parsing (for a larger overview cf. Bansal et al. (2014, p. 809)).

Naturally, the main goal of using word representations is adding further information to a classification task. How to make this information maximally relevant depends on the task given. Thus, whether we want two words to be considered similar depends on the task in which they are being classified (cf. Guo et al. (2014)). For example, Bansal et al. (2014) train their word representations on dependency context instead of raw linear context. In the following we will apply word vectors to the task of NER.

Vector based word representations have a successful history of use in information retrieval and computational semantics as an implementation of the long-standing linguistic hypothesis that words that occur in similar contexts tend to have similar meanings (Harris, 1954). More recently, word vectors have also been shown to be able to capture linguistic regularities of both semantic ('king' to 'man' is like 'queen' to 'woman') and syntactic nature ('ran' to 'run' is like 'laughed' to 'laugh') (Mikolov et al., 2013b).

We first discuss our method of extracting word vectors and adding them to the classification task before describing the task of NER and our more experiments more concretely. In the final discussion we primarily explore engineering options related to the incorporation of the word embeddings and the size of the unlabelled data set.

2 Extracting the word vectors

The method used to extract word vectors, *word2vec*, implements two models that take tokenised but otherwise non-processed text and derive a feature vector for every type in this data set. For this paper we used the continuous skip-gram model, a neural network model that avoids multiple hidden layers in order to allow extremely fast and efficient training, for example when compared

to most clustering algorithms. During training, each word in the data set is used as an input to a log-linear classifier, which learns word representations by trying to predict words occurring within a certain range to either side of the word.[1] As those words occurring further away from the input word are less likely to be related to it, these words are given less weight (cf. Mikolov et al. (2013a, p. 4f.) for the original presentation).

Having chosen a word representation, an almost equally important choice regards the method by which the chosen feature is incorporated into a linear model. Using *word2vec* embeddings for syntactic parsing, Bansal et al. (2014) report that simply adding the relevant word vector to the feature vector during training does not yield improved results. One solution presented for this problem is clustering the word vectors (Guo et al., 2014). We adopt this solution using choose k-means clustering and introduce the clusters into the classification problem by adding a further feature representing the cluster of each word.

3 Named Entity Recognition

NER is a sequence prediction problem. Given a tokenised text, the task is that of predicting which words are locations, organisations or persons. To be able to distinguish multiple adjacent instances of the same type of named entity and a named entity spanning multiple words, a beginning-inside-outside (BIO) encoding is used (Sang and Veenstra, 1999).

Both training and testing data for the classifier were taken from the annotated CoNLL03 corpus (Sang and De Meulder, 2003). This data, which is a collection of news wire articles from the Reuters Corpus, is annotated with part-of-speech (POS), syntactic chunk and named entity tags. The features we extracted given one word in the data are: (1) tokens plus their POS tags in a window of ± 2 (2) token's syntactic chunk in a window of ± 1 (3) upper-cased tokens in a window of ± 2 (4) initial capitalization pattern of tokens in a window of ± 2 (4) the previous two predicted tags (5) the conjunction of the previous tag and the current token (6) prefixes and suffixes of the token (7) more elaborate word type information for the token (as employed by Zhang et al. (2003)).

A very simple implementation of evaluation was allowed by the CoNLL03 scoring method which evaluates whether the NER system correctly identified a full named entity. Furthermore, the evaluation script gives a clear presentation of performance in the different categories (person, location, organisation, miscellaneous), though we do not discuss the potential reasons for variations in performance in these categories here.

4 Experimental setup

All of our experiments were based on the RCV1 corpus which contains one year of Reuters English newswire from August 1996 to August 1997 (Lewis et al., 2004). Preprocessing of the RCV1 corpus involved the extraction of text from the news files as well as sentence and word tokenization, both of which we did using the NLTK toolkit.

In order to evaluate the effect of the size of the non-labelled corpus on performance we trained word embeddings on different subsets of RCV1. The smallest subset was the CoNLL03 corpus. Furthermore, we trained *word2vec* models on a quarter, half and three quarters of RCV1. In Table 1 below we give a rough estimate of the number of documents used for training each model (where a document contains between a few hundred and several thousand words) as well as the number of words represented with word vectors in the *word2vec* model. As could be expected given the Zipf distribution of words, the number of unique types does not grow linearly with the number of tokens in a model but begins to stagnate at a large number of documents.

This effect is fortified by necessary design choices: While training the CoNLL03 corpus we set the *word2vec* 'min_count' variable to one (which means that all tokens will be considered) whereas for the larger data sets it was set to the default value of five (only words occurring at least five times are represented) to reduce processing costs.

word2vec was reimplemented for use in Python by Rehurek and Sojka (2010). We use this implementation to build our models, using the default setting for vector dimensionality (100), as well as for the other parameters such as the number of training iterations and the size of the window.

[1] Bansal et al. (2014) report that the size of this range or window has a significant impact on the resulting word vectors. Large windows result in more semantically accurate groupings, whereas smaller windows result in the grouping of words with similar part-of-speech tags. Exploring various window magnitudes was unfortunately not within the scope of this paper.

data set	nr types	nr documents
CoNLL03	30 290	1 393
1/4 RCV1	145 824	∼ 200 000
1/2 RCV1	221 066	∼ 400 000
3/4 RCV1	289 345	∼ 600 000
RCV1	356 843	∼ 800 000

Table 1: Data set statistics.

The classification itself is a greedy implementation of the Linear Support Vector Classification algorithm as implemented in the scikit-learn software, using the default values (Pedregosa et al., 2011). Linear SVC was shown by Ratinov and Roth (2009) to perform comparably to more computationally complex and costly search algorithms such as beamsearch or Viterbi.

5 Discussion

During the following discussion of results when referring to performance we are referring to performance as indicated by the overall F-measure returned by the CoNLL03 evaluation script.

5.1 Cluster granularity

A practical challenge in generating clusters from word embeddings lies in choosing the relevant cluster granularity, i.e. the cluster granularity that maintains the information relevant to the classification task at hand.

Given the considerable time demands of generating clusters we performed all of our initial granularity experiments on the *word2vec* embeddings constructed from the smallest dataset, CoNLL03. Through our first experiment, we attained a rough idea of a task-adequate dimension of cluster granularity: We evaluated three dimensions of clusters (100, 1000 and 5000) extrinsically by considering their effect on the performance of the NER system. Granularity 1000 performed best, suggesting that this is the correct range of dimensionality.

In a next step, we manually inspected the clusters built at this granularity (again from the CoNLL03 *word2vec* model). Though they were rather noisy (e.g. numbers were included in almost every cluster[2]), they seemed to capture some regularities. In Example 1 we give two excerpts from clusters from this dataset, where the first is primarily a collection of person names (and some company names) and the second a collection of city names. Clusters at granularity 1000 from the larger datasets were similarly noisy.

(1) a. 0-6 1-15 1-7 1.4871 ... Alexia Angelica ... Jill Jimy Jolene Juliet KTM Kandarr ... Yamaha Yi Zina Zrubakova

b. AMSTERDAM ANKARA Auth ... VIENNA WARSAW WELLINGTON WINNIPEG

A solution to the cluster granularity problem proposed in many papers (e.g. Miller et al. (2004)) is the combination of multiple granularities, for example through hierarchical clustering. Surprisingly, for training on the CoNLL03 corpus the combination of different cluster granularities (we tried various combinations of 500, 1000 and 1500) did not improve accuracy. To ascertain the validity of this finding for all of our word embedding models, we repeated this experiment on the *word2vec* model built using half of the RCV1 corpus. Here, we found considerable improvement in performance, with a growth in performance for every added granularity (cf. Table 2), suggesting that further improvement could be achieved with an even greater number of clusters.[3]

granularity	performance
1500	82.83%
1000 + 1500	83.52%
500 + 1000 + 1500	83.81%

Table 2: Testing granularity with 1/2 RCV1.

5.2 Unlabelled corpus size

The second question we aimed to elucidate in this paper is what effect the size of the unlabelled corpus has on the performance of the NER system. Given our experimental set-up, this essentially boils down to the following question: Given an unlabelled corpus that is both periodically and stylistically similar to the testing data can we expect better performance the larger the corpus?

It is important to note that in particular in our experiments the size of the unlabelled corpus does

[2]To reduce this kind of noise, Turian et al. (2010) preprocess the data by removing all sentences that are less than 90% lowercase a-z (not counting whitespaces). In previous experiments we did not find this measure to improve performance.

[3]Limited processing power detained us from putting this hypothesis to the test.

not have a direct correlation to the percentage of word types occurring in the testing data that are also covered by the word embeddings model. This can be explained with reference to Table 1 and the setting of the *word2vec* 'min_word' variable discussed above. Whether choosing to train the model on all word types occurring in the training data as well as additional unlabelled data will improve performance by preventing the possibility of unknown words.

We did not find evidence that suggests a direct correlation between corpus size and performance. Rather, the improvement stagnated at around half of the RCV1 corpus. All results are given in Figure 1 and compared to the no cluster baseline; the F-measures given are achieved from adding clusters at granularity 1000, built from *word2vec* models trained on the various data sets, to the NER classifier.

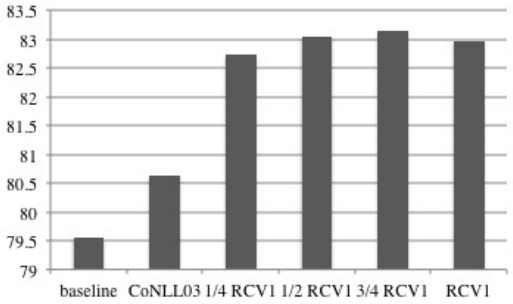

Figure 1: F-measures given in percent.

One possible explanation for the stagnating performance of the larger data set is that other training settings need to be employed for optimal training (e.g. higher vector dimensionality or more training iterations). Regardless of the validity of this explanation, the results suggest that optimal training of a word embeddings model for a certain task is a more complex problem than training it with the maximum amount of data.

6 Conclusion

In conclusion, we would like to present a summary of our results: In Figure 2 we show the Linear SVC baseline (a) in comparison to our lowest and our best performing experiment at cluster granularity 1000 (CoNLL03 and ¾ RCV1, (b) and (c), respectively). Finally, we show the possible improvement through combination of multiple granularities by comparing the results of using ½ RCV1 with just 1000 clusters (d) and combining this granularity with 500 and 1500 (e).

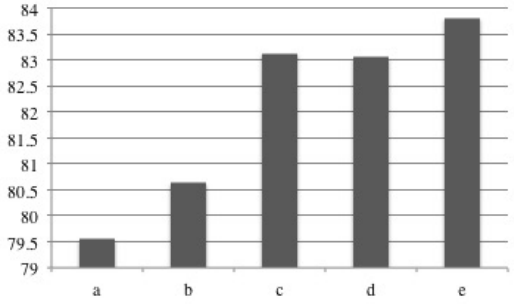

Figure 2: F-measures given in percent.

The graph reemphasises the two key observations discussed above:

1. Performance of the NER model improved with growth of the size of the unlabelled data set but only to a limit (here at around 300 000 types) at which it even started to drop.

2. Combining multiple cluster granularities led to our best improvement. It did not improve performance for smaller data sets.

We believe these findings adequately illuminate the complexity of optimally enhancing NLP tasks with unsupervised word representations of any kind.

There are naturally a number of ways this project could be replicated in a more sophisticated way to yield a yet more sophisticated understanding and therewith likely further gains in performance. For one, the performance by named entity class is potentially helpful data that was not considered here. For example, for result (e) in Figure 2 above, class-specific F-measures ranged from 74.77% (miscellaneous) to 91.47% (person). Interestingly, for the miscellaneous class few results fell over the baseline (74.02%) whilst this was the case for all results for the person class (baseline at 86.27%).

A further question worth exploring is how similar the unlabelled data needs to be to the testing data to achieve good results. Our data was from the same time period (important for named entities) and the same domain (newspaper articles). Exploring how much this can be altered to nevertheless maintain good results would be a valuable question to answer for insights on optimal training of word representation models.

Acknowledgments

This article is the result of course work produced during the Master's course in Language Technology at Gothenburg University for the Swedish Research Council's project 2012-5738 ("Towards a knowledge-based culturomics"). I would like to thank the Center of Language Technology for its generous funding towards my attendance at the 2015 NoDaLiDa conference.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 110–120.

Zellig Harris. 1954. Distributional structure. *Word*, 10:146–162.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of Human Language Technologies*, pages 337–342.

Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandra Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, Conference on Natural Language Learning 2009, pages 147–155.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of Language Resources and Evaluation Conference 2010 workshop New Challenges for NLP Frameworks*, pages 46–50.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conference on natural language learning-2003 shared task: Language-independent named entity recognition. In *Proceedings of Conference on Natural Language Learning-2003*, pages 142–147.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, pages 173–179. Bergen, Norway.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics 2010, pages 384–394.

Tong Zhang, Fred Damerau, and David Johnson. 2003. Updating an NLP system to fit new domains: an empirical study on the sentence segmentation problem. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Conference on Natural Language Learning-2003*, pages 56–62. Edmonton, Canada.